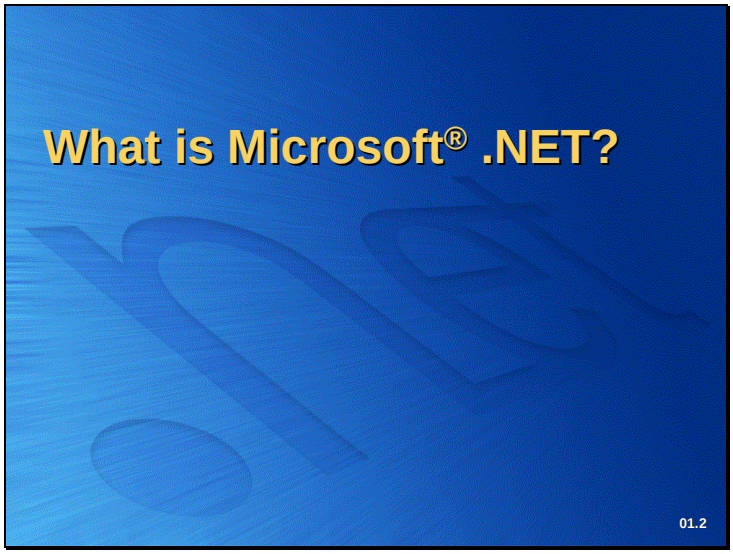
Microsoft
.net developer
training
Create XML Web Services and Applications with Visual Studio .NET

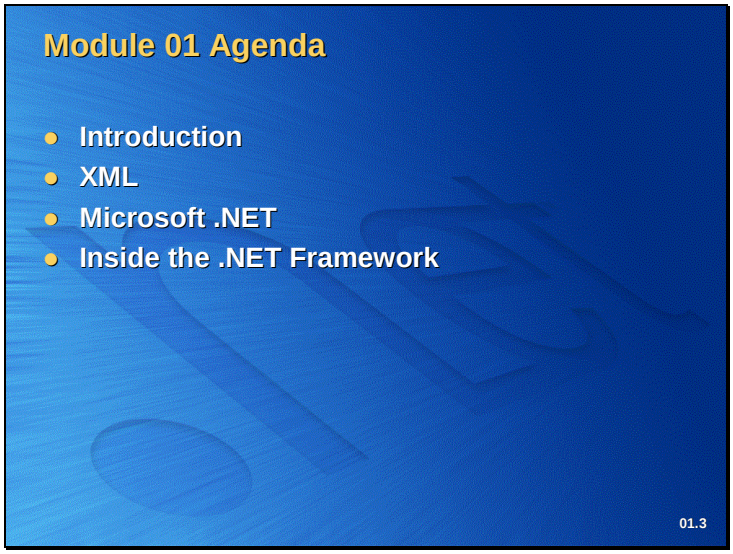


What Is Microsoft® .NET?

Agenda: Day 1

- ▶ Module 01: What is Microsoft® .NET?
- Module 02: Visual Studio® .NET
- Module 03: The .NET Languages
- Module 04: Building ASP® .NET Applications
- Module 05: Using Data Access in .NET Applications
- Module 06: Object Oriented Concepts in .NET
- Module 07: Creating Windows Applications
- Module 08: Using XML in .NET

 ©2002 Microsoft Corporation



Module 01 Agenda

What you will learn:

- What is .NET?
- How it works
- What it can do for you?

Introduction

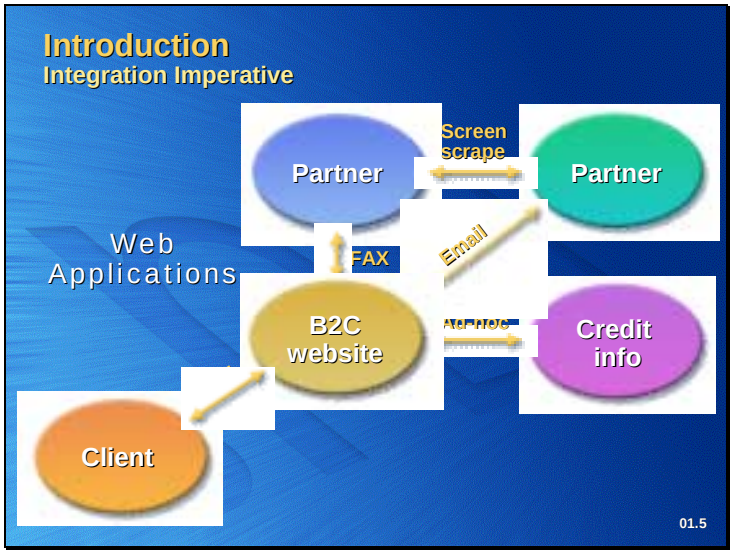
Technology Drivers

Imagine you want to turn up the heat in your house while away from home. You could use a .NET experience that controls your household utilities through a Smart Device, which could be your office desktop computer, your smart phone or a kiosk used while shopping for dinner ingredients.

There already is a refrigerator that has a Smart Device built-in component that includes a bar code reader. This Smart Device can be set to keep the refrigerator stocked by automatically ordering grocery items at user set intervals.

For more information, see http://www.microsoft.com/net/qa.asp

 ©2002 Microsoft Corporation



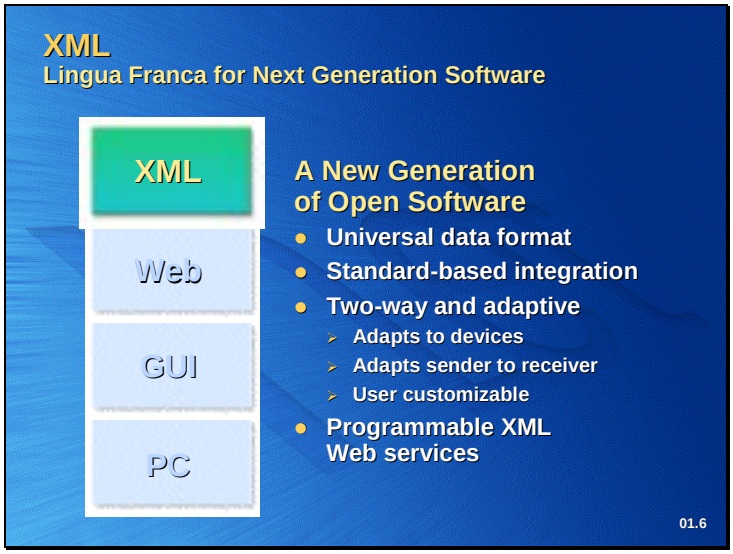
Introduction

Integration Imperative

Most Web Applications have at least one of the above links, but also the common problems that come with these applications. Historically, these links are add-ons, afterthoughts that connect islands of technology. These links are custom-coded and ad-hoc making them un-scalable and fragile. Consequently, the integration that virtual organizations depend on is too difficult, expensive, slow and rigid.

Business to Customer (B2C) scenarios require developers to create solutions by using a variety of technologies. These solutions may also involve **Business-to-Business** applications. These solutions evolve into constellations of multiple applications, devices and services.

The Web addressed the issue of how users interact with applications, but it failed to solve the dilemma of how applications interact, even though this is core problem in business integration today. Traditional Enterprise applications must be integrated with external partners who have their own Intranets powered by different technologies. [Recent study indicates that 60% of e-business transactions are still printed to paper for processing.]



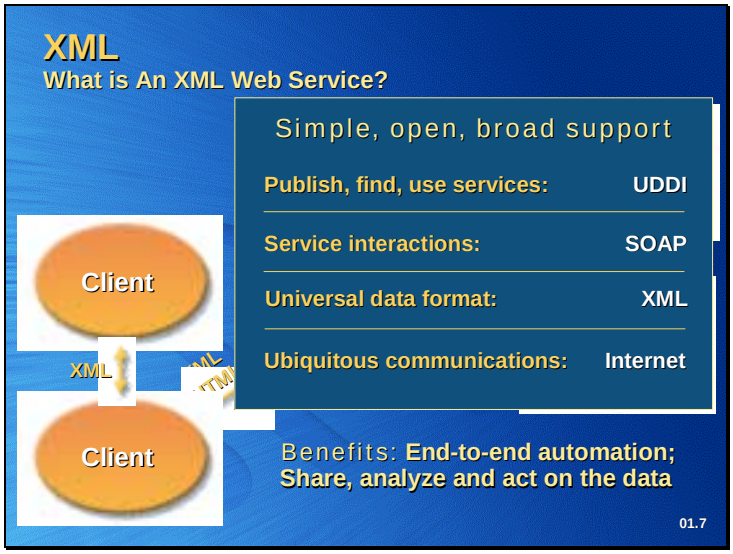
XML

Lingua Franca for Next Generation Software

XML is a standards-based data format that can be used to transfer data across the Internet. It does not matter what systems, machines or applications are communicating. XML abstracts those differences out of the equation.

The .NET Framework uses XML extensively—in configuration files, XML Web Services, as ADO.NET DataSets etc.

 ©2002 Microsoft Corporation



XML

What Is An XML Web Service?

An **XML Web service** is a unit of application logic that provides data and services to other applications. Applications access XML Web services by using standard Web protocols and data formats such as HTTP, XML and SOAP, independent of how each XML Web service is implemented.

XML Web services combine the best aspects of component-based development and the Web and are a cornerstone of the Microsoft .NET programming model. XML Web services transform read-only Web sites into computing sites that can expose methods and/or consume other XML Web services.

Using XML, you can share data between any operating system or network and can expose existing code as XML Web services, thereby connecting disparate systems. Universal Description Discovery and Integration, UDDI specifications define a standard way to publish and discover information about Web services.

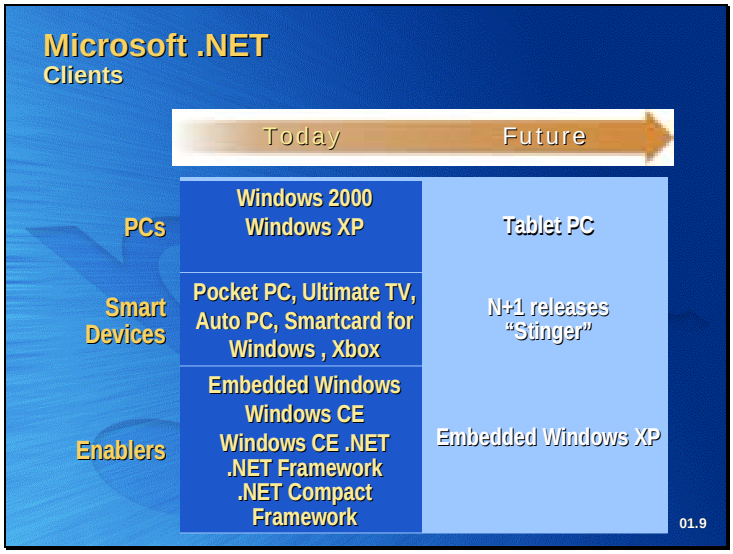
Microsoft .NET

Building On an XML Foundation

XML is an industry standard and there currently is an XML Kernel in all Microsoft products, especially the latest generation of servers. The .NET Framework is built into all the .NET products, that is Visual Studio .NET, Visual Basic .NET, Visual C#, Visual J# and so on. The .NET Framework soon will be integrated into all Microsoft products, such as Microsoft Office .NET and the new Enterprise servers, and it will be tightly integrated Microsoft tools and third party products.

Microsoft's common tools and programming model lets you leverage your existing code and skills while the .NET Framework allows your applications to talk to a wide variety of clients without writing a separate application for each interface you support.

 ©2002 Microsoft Corporation

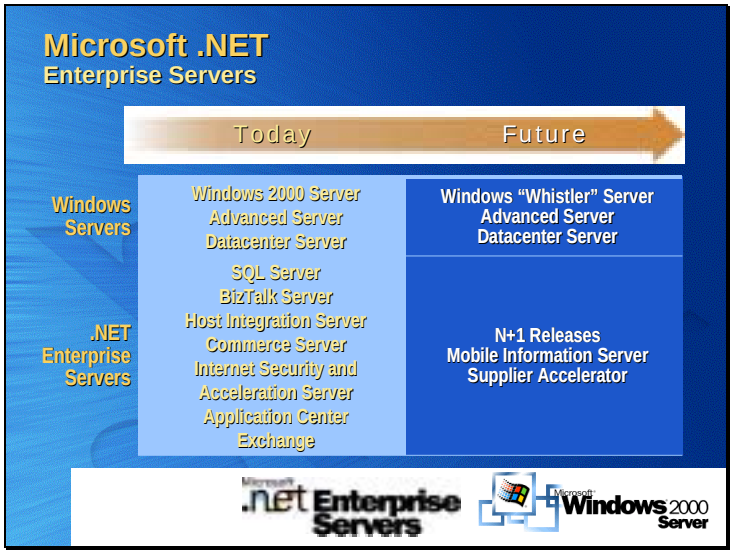


Microsoft .NET

Clients

Microsoft .NET, supports a number of different clients, from Windows CE devices to PCs running Windows 2000. The PC is also a Smart Device, the ultimate Smart Device.

Microsoft is working on the following software products for Smart Devices: Windows XP, Windows Me, Windows CE, Windows Embedded, the .NET Framework and the .NET Compact Framework.



Microsoft .NET

Enterprise Servers

The current generation of Microsoft Enterprise servers offer a high level of interoperability with .NET and extensive XML support. The next generation of servers will be completely integrated.

.NET Enterprise Servers

Microsoft .NET Enterprise Servers

- Integrated suite for operating, managing and orchestrating Web Applications and services
- Enterprise reliability and scalability for the most demanding solutions
- Foundation for the next generation of business applications and services
- Scale out hardware performance, availability and economics
- Extensive XML support for integration within and across organizations

 ©2002 Microsoft Corporation

Microsoft .NET

Building Block Services

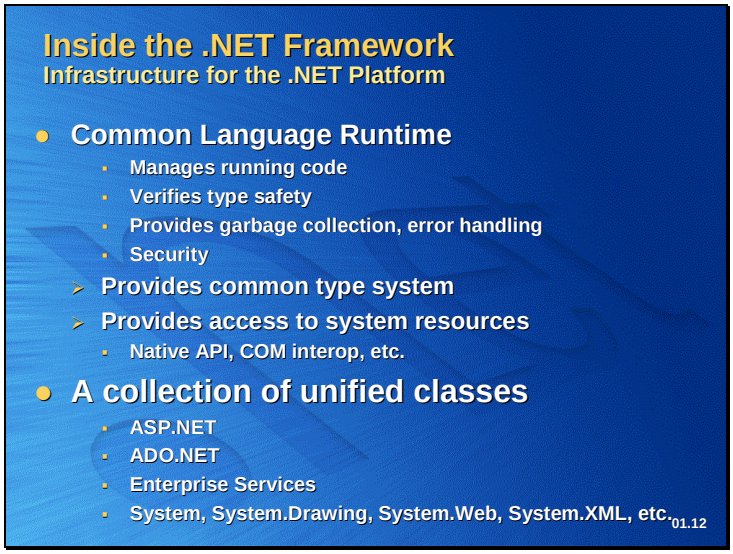
Passport is one of the 10 largest Web sites in the world with 160 million-plus active accounts, and it is increasing at a rate of more than 10 million a month. Though Passport receives more than 1.5 billion authentications each month, the Web site itself seldom receives visitors because it is an XML Web service.

My Services is a set of user-centric XML Web services that puts users in control. It manages and protects user info and offers consistent and personalized access from any application, device, service or network. My Services' platform approach creates incredible industry opportunities. Users do not generally interact with the service directly except for administrative functions, but access happens through the different applications, devices and services. It helps manage, protect and securely share data, e.g. calendar: How to deliver information to a PC or a cell phone? How to share a subset of your calendar with co-workers, friends and family or your dentist?

My Services interacts via open protocols:

- Any OS, language or network
- Windows, Windows CE, UNIX, Mac, Palm, etc.
- No Microsoft software required

My Services include: **myProfile, myAddress, myApplicationSettings, myDevices, myDocuments, myFavoriteWebSites, myLocation, myWallet, myNotifications, myCalendar, myContacts and myInbox.**



Inside the .NET Framework

Infrastructure for the .NET Platform

The **.NET Framework** is the infrastructure for the .NET Platform. The .NET Framework is a computing platform designed to simplify application development in the highly distributed environment of the Internet. The .NET Framework has two main components, the CLR and the .NET Framework class library.

The **CLR** is the foundation of the .NET Framework. Think of the Common Language Runtime as an agent that manages code at execution time, providing core services such as memory management, thread management and remoting, while enforcing strict safety and accuracy of the code. In fact, the concept of code management is a fundamental principle of the CLR. Code that targets the CLR is known as managed code while code that does not target the Runtime is known as unmanaged code.

The **Common Type System** defines how types are declared, used and managed in the CLR. It also is an important part of the Runtime's support for cross-language integration.

The Common Type System performs the following functions:

- Establishes a framework that enables cross-language integration, type safety and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow to help ensure that objects written in different languages can interact.

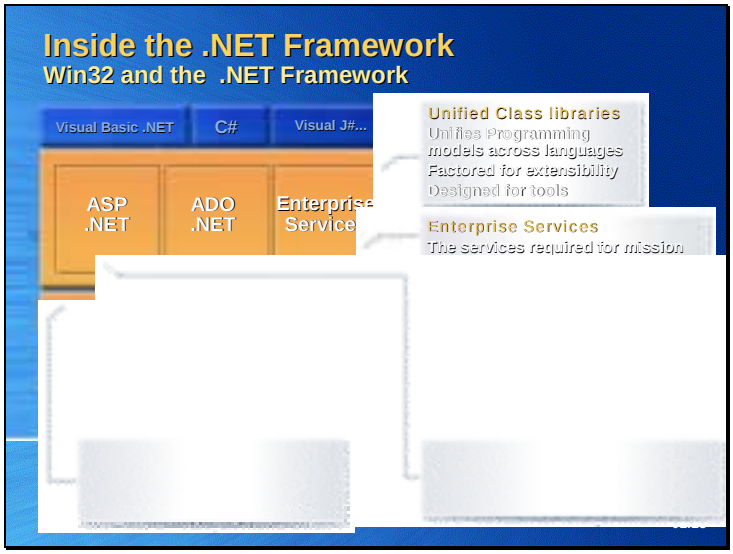
More information on CLR is discussed later in this module.

The **.NET Framework Unified Class Library** is a comprehensive, object-oriented collection of reusable classes that will help you develop applications ranging from traditional command-line or Graphical User

 ©2002 Microsoft Corporation

Interface (GUI) applications to applications based on the latest innovations provided by ASP.NET and Web services.
Most of the members defined by types in the .NET Framework Class Library are CLS compliant, though some types in the class library have one or more members that are not CLS compliant. The .NET Framework also provides several Runtime hosts, which are unmanaged components that load the CLR into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework SDK not only provides several Runtime hosts, but also supports the development of third-party Runtime hosts.

For example, ASP.NET hosts the Runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the Runtime to enable .aspx pages and Web services. Internet Explorer is an example of an unmanaged application that hosts the Runtime (in the form of a MIME type extension). Using Internet Explorer to host the Runtime enables you to embed managed components or Windows Forms controls (the .NET version of a Microsoft® ActiveX control) in HTML documents.



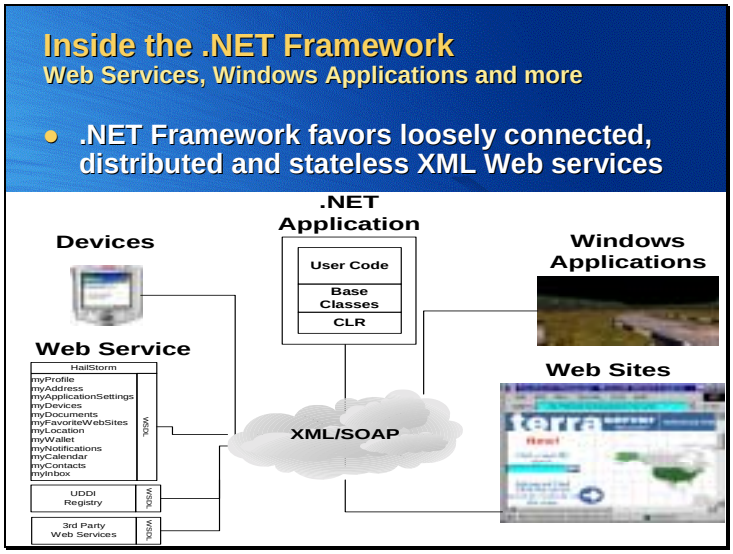
Inside the .NET Framework

Win32 and the .NET Framework

The .Net Framework is an environment for building, deploying and running XML Web services and other applications. This illustration shows how the .NET Framework (shown in orange) fits into the .NET Platform.

The CLR makes it easy to design components and applications whose objects interact across languages. Objects written in different languages can communicate and their behaviors can be tightly integrated. For example, you can define a class and then use a different language to derive a class from your original class or call a method on the original class. You also can pass an instance of a class to a method on a class written in a different language. This cross-language integration is possible because language compilers and tools that target the Runtime use a common type system defined by the Runtime, and they follow the Runtime's rules for defining new types as well as for creating, using, persisting and binding to types.

 ©2002 Microsoft Corporation



Inside the .NET Framework

Web Services, Windows Applications and more

Microsoft .NET is built from the bottom using open architecture and it provides a robust, open platform for developing the next generation of Web Applications, XML Web services and Windows Applications.

You can create Internet-ready distributed applications by using the .NET Framework. A traditional, tightly connected and stateful model fails to work on the Web. ASP.NET provides that infrastructure with built-in session state functionality (thanks to the .NET Framework) that enables you to:

- Automatically identify and classify requests coming from a single browser client into a logical application session on the server
- Store session-scoped data on the server for use across multiple browser requests
- Raise appropriate session lifetime management events (**Session_OnStart**, **Session_OnEnd** and so on) that can be handled in application code
- Automatically release session data if the browser does not revisit an application within a specified time-out period

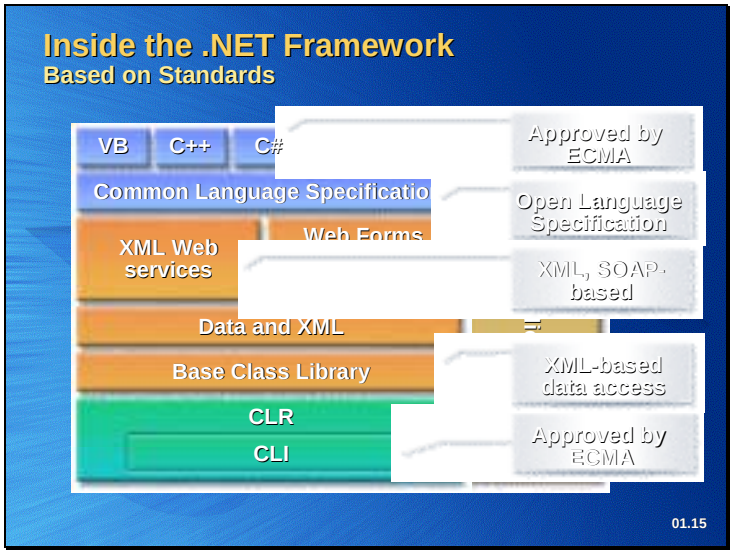
The .NET Framework supports Windows® Applications. Windows Applications have been expensive to build and deploy, especially when compared to the cost of building and deploying Web Applications. In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a Rapid Application Development (RAD) environment such as Microsoft Visual Basic®. The .NET Framework incorporates aspects of existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars and other screen elements with the flexibility necessary to accommodate shifting business needs. In fact, you use the same forms designer to build a Windows Form as you do a Web Form in Visual Studio .NET. So, .NET

Windows Applications are easier to develop than traditional Windows Applications and with auto-deployment, the cost of deploying the application is similar to the cost of deploying a Web Application.

This topic is discussed in greater detail in the module on Windows Applications.

 ©2002 Microsoft Corporation



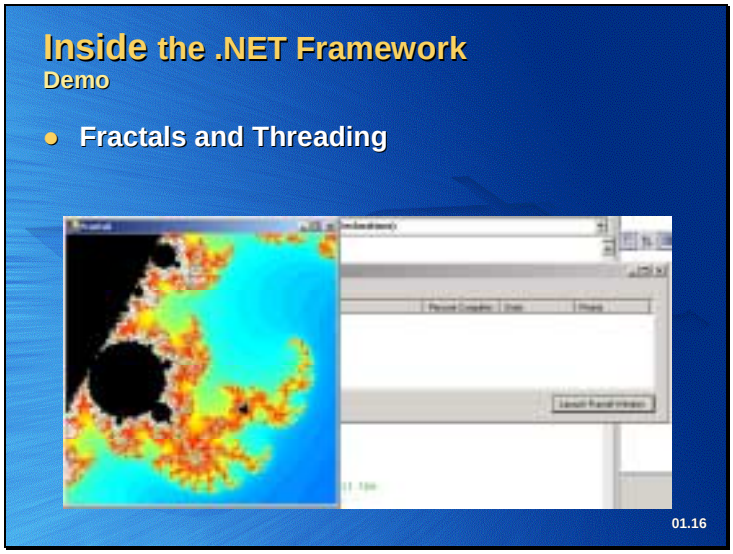
Inside the .NET Framework

Based on Standards

Microsoft .NET is built on standards. Both the Common Language Infrastructure (CLI) and Visual C # are approved by the European Computer Manufacturers Association (ECMA) as standards. The data and Web services are built on XML and SOAP.

.Net Framework supports Web standards:

- **HTML**- Hyper-Text Markup Language
- **XML** - Extensible Markup Language, the universal format for structured documents and data on the Web
- **WSDL-** Web Services Description Language (WSDL)
- **SOAP -** Simple Object Access Protocol, a simple, XML-based protocol for exchanging structured and type information on the Web. The protocol contains no application or transport semantics, which makes it highly modular and extensible
- **XPATH -** XML Path Language
- **XSLT -** The successor to Extensible Stylesheet Language (XSL), XSL Transformations (XSLT), is an XML-based language that enables you to transform one class of XML document to another



Inside the .NET Framework

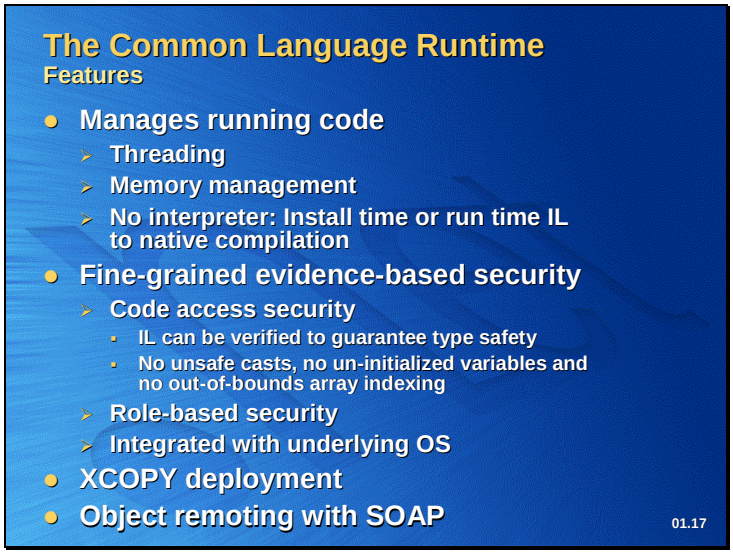
Demo

The purpose of this demo is to illustrate how the Framework provides:

- Threading
- Windows Forms
- Graphics support (Fractals)

Threading is a feature available to all languages in a standard way via the .NET Framework.

 ©2002 Microsoft Corporation


The Common Language Runtime
Features
Manages running code
Threading
Memory management
No interpreter: Install time or run time IL to native compilation
Fine-grained evidence-based security
Code access security
IL can be verified to guarantee type safety
No unsafe casts, no un-initialized variables and no out-of-bounds array indexing
Role-based security
Integrated with underlying OS
XCOPY deployment
Object remoting with SOAP
01.17

The Common Language Runtime

Features

The Common Language Runtime (CLR)

The execution engine for .NET Framework applications. It provides a number of services including:

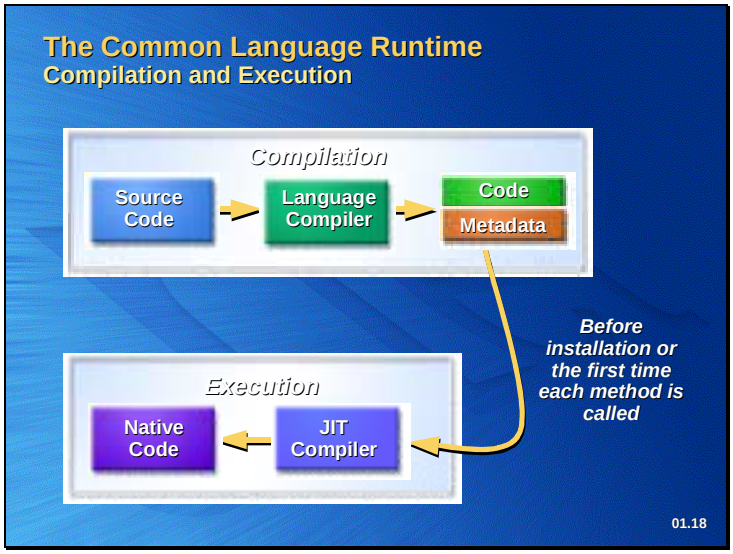
- Code management (loading and execution)
- Application memory isolation
- Verification of type safety
- Conversion of IL to native code
- Access to metadata (enhanced type information)
- Managing memory for managed objects
- Enforcement of Code Access Security
- Exception handling including cross-language exceptions
- Interoperation between managed code, COM objects and pre-existing DLLs (unmanaged code and data)
- Automation of object layout
- Support for developer services (profiling, debugging, etc.)

Native Code
The code compiled to processor-specific machine code.

Managed Code
Managed code runs under a "contract of cooperation" with the CLR and it must supply the metadata necessary for the CLR to provide services such as memory management, cross-language integration,

Code Access Security and automatic lifetime control of objects. All code based on Microsoft Intermediate Language (MSIL) executes as managed code.

Microsoft Intermediate Language (MSIL) is used as the output of a number of compilers and as the input to a Just-In-Time (JIT) compiler. The CLR includes several JIT compilers for converting MSIL to native code.



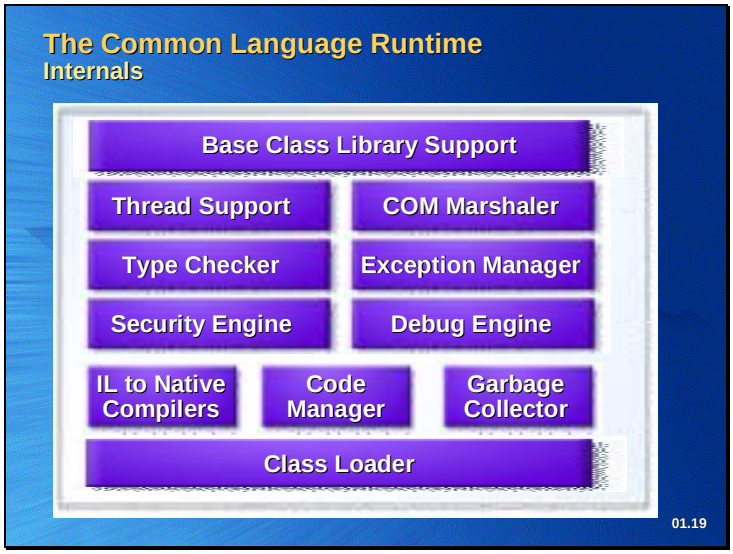
The Common Language Runtime

Compilation and Execution

The above diagram illustrates the process used to compile and execute managed code, or code that uses the CLR. Source code written in Visual C#, Visual Basic .NET or another language that targets the CLR is first transformed into MSIL by the appropriate language compiler. Before execution, this MSIL is compiled by the Just-in-Time (JIT) complier into native code for the processor on which the code will operate. The default is to JIT compile each method when it is first called, but it is also possible to “preJIT” MSIL code to native code at assembly install-time. With this option, all methods are compiled before the application is loaded so as to avoid the overhead of JIT compilation on each initial method call. You use the Native Image Generator (Ngen.exe) to create a native image from a managed assembly and install it into the native image cache. Once the code is compiled and cached, the CLR does not need to re-compile the MSIL code until the assembly is updated, the Just-In-Time compiled code is removed from the cache, or the machine is restarted.

A point worth noting is that all languages targeting the CLR should exhibit a similar performance. While some compilers may produce better MSIL code, large variations in execution speed are unlikely.

 ©2002 Microsoft Corporation



The Common Language Runtime

Internals

This slide gives an idea of what the CLR incorporates. It isn't necessary to understand what these components do since you may never see them, but below is a brief description of the CLR internals. Support for **Base Class Libraries**: Reflection, Root Object type, Floating Point, etc.

The .NET Framework provides a run-time environment called the CLR, which manages the execution of code and provides services that make the development process easier. Compilers and tools expose the runtime's functionality and enable you to write code that benefits from this managed execution environment. Code developed with a language compiler that targets the runtime is called managed code and it benefits from features such as cross-language integration, cross-language exception handling, enhanced security, versioning and deployment support, a simplified model for component interaction as well as debugging and profiling services. It interacts with COM and supports COM Marshaling and multi-threading.

You can use a custom Marshaler to precisely control the way in which a managed type is exposed to COM or an unmanaged type is exposed to the .NET Framework. You also can use it in place of the Interop Marshaler to manipulate the marshaling process.

The Marshaler provides a bridge between the functionality of old and new interfaces. Custom marshaling enables clients you designed to work with an old interface to also work with servers that only implement a new interface and it enables clients built to work with a new interface to work with servers that expose the old interface.

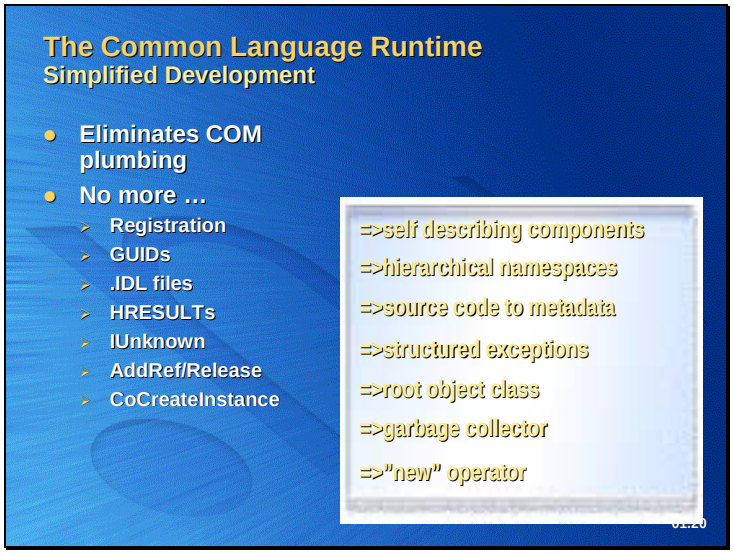
Class Loader - At the bottom is the Class Loader, which manages the loading of classes and metadata. In addition to the Class Loader, CLR involves CPU independent IL to native code compilers, the Code Manager that controls execution and the **Garbage Collector** that provides automatic memory management for all objects.

Note: Deterministic Finalization refers to the termination behavior of objects in Visual Basic/COM, or even C++ classes for that matter. When the last reference to a COM object is released, the object is immediately terminated. The Class_Terminate procedure is called in Visual Basic classes, and all resources that it has are released. However in the .NET world, after objects are released they don't get "cleaned up" until the garbage collector runs and finds them. The time it takes for the garbage collector to run depends on a lot of factors like current load, memory usage, etc. Some developers relied on deterministic finalization to automatically clean up their resources after they release their reference. You could set a data object to nothing and all of its open database connections and RecordSets would be released without having to explicitly call Close on them. In .NET, developers should remember to call Dispose (or Close) on objects that hold expensive resources before releasing the object reference.

More info on COM Marshaler is available at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguidnf/html/cpconcustommarshaling.asp

 ©2002 Microsoft Corporation

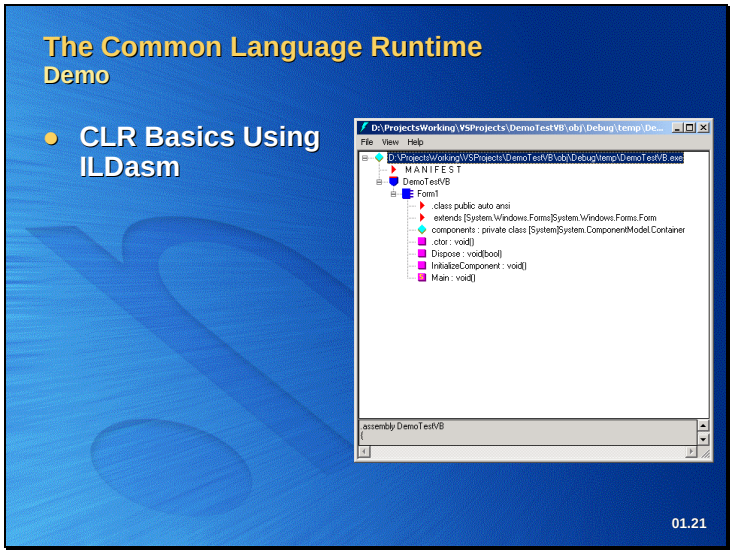


The Common Language Runtime

Simplified Development

One of key design goals of the CLR was to simplify development. The CLR in .NET Framework eliminates the plumbing previously necessary for developers to write COM components. For example:

- Registration is unnecessary because applications can be self-described
- GUIDs are unnecessary because classes are organized in a hierarchical naming scheme with human readable names
- IDL files, or the programming languages that natively understand the object model the CLR provides, are unnecessary because the system and languages have the same representations
- HRESULTs are unnecessary because all .NET languages support structured exception handling
- IUnknown is unnecessary since its functionality is provided by the root object
- AddRef/Release is unnecessary because .NET provides automatic lifetime management through garbage collection
- CoCreateInstance becomes the new operator



The Common Language Runtime

Demo

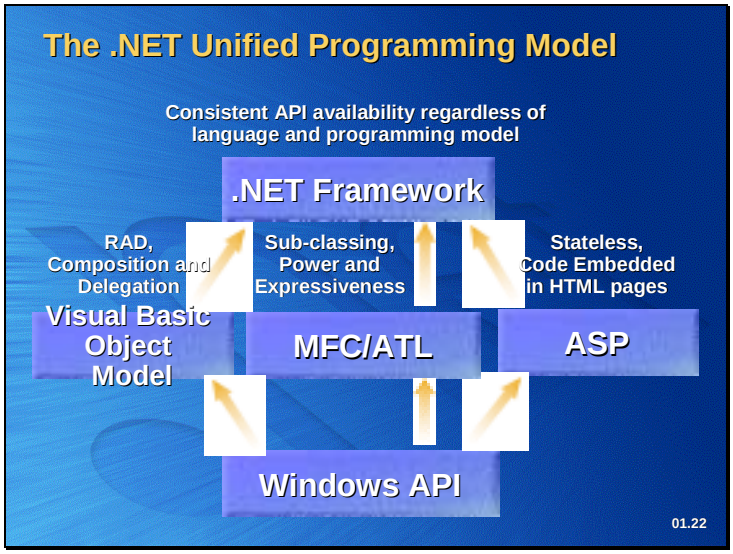
The purpose of this demo is to illustrate the opening assembly with ILDasm utility

Explain how:

- Meta data is stored in assembly
- MSIL is what is executed and is part of assembly
- .NET Framework provides utilities for working with assemblies

The ILDasm.exe tool parses any .NET Framework .exe or .dll module, and shows the information in a human-readable format. ILDasm.exe shows more than just the Microsoft intermediate language (MSIL) code—it also displays namespaces and types, including their interfaces. You can use ILDasm to examine native .NET Framework modules, such as Mscorlib.dll, as well as .NET Framework modules provided by others or created yourself. Most .NET Framework developers will find ILDasm indispensable.

 ©2002 Microsoft Corporation



The .NET Unified Programming Model

The .NET Framework unifies various programming models. Let us go back a bit in history and describe some of the disparate programming models that have been very popular and that people use today.
In the good old days of early Windows, you had the Windows API. To write applications you fired up your C compiler, #included Windows.h, created a Windows procedure, handled your Windows messages – basically the old Petzold style of Windows programming. While this worked, it was neither productive, nor was it easy.

Over time, various programming models on top of the windows API have emerged.

Visual Basic - RAD…instantiate a form, drag components onto the form, write event handlers and through delegation your code executes.

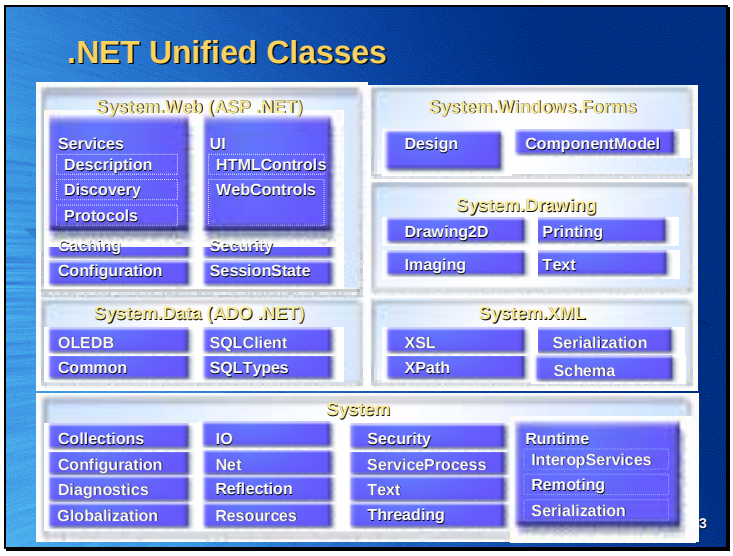
MFC/ATL took a different view. The key concept here is sub-classing. You create a subclass from an existing monolithic, object oriented framework. While this gives you more power and expressiveness, it doesn't match the ease or productivity of Visual Basic's composition model.

ASP - With the Web, we've seen the emergence of the ASP model where you write stateless code that's embedded in HTML pages.

If you look at the slide above, one of the problems is that your choice of the programming model unfortunately becomes your choice of programming language. If you are a skilled Microsoft Foundation Class (MFC) developer, and you need to write some code in an ASP page, your skills will fail to translate. Likewise, if you know more about Visual Basic, there's not much that transfers to MFC.

There also is not a consistent availability of API. Each of these models has there own solutions to a number of core problems that are common to all of the models. For example, how do I deal with File I/O? How do I perform string formatting? How do I handle security, threading, etc?

What the .NET Framework does is unify all of these models, providing a consistent API available everywhere regardless of the language you use or what programming model you are targeting.



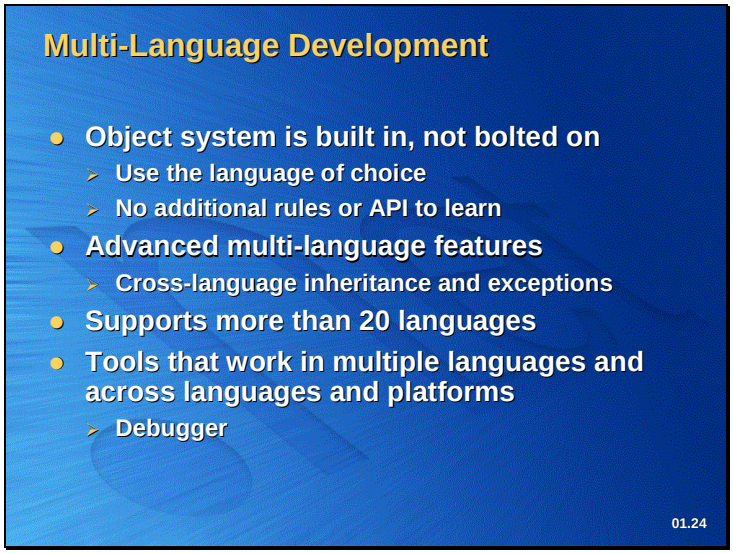
.NET Unified Classes

The **.NET Framework** contains a set of unified classes that provide services to all .NET languages. As a result there now are no second-class languages. All .NET languages are fully object-oriented and can extend the functionality of the .NET Framework as needed.

The slide shows a number of the key classes in the .NET Framework, but it is by no means a complete list. The **.NET Unified Classes** provide a rich set of features that:

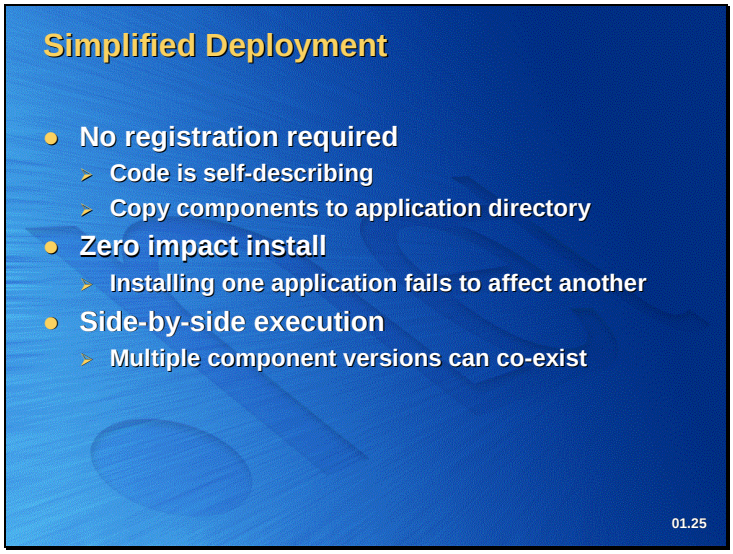
- Increase developer productivity by supplying a single hierarchical and intuitive programming model
- Enable cross-language inheritance and debugging
- Make it easy to add or modify .NET Framework features
- Allow creation of secure applications
- Integrate well with tools

 ©2002 Microsoft Corporation


Multi-Language Development
Object system is built in, not bolted on
Use the language of choice
No additional rules or API to learn
Advanced multi-language features
Cross-language inheritance and exceptions
Supports more than 20 languages
Tools that work in multiple languages and across languages and platforms
Debugger
01.24

Multi-Language Development

The supported languages include: Perl, Python, COBOL, Jscript®, Eiffel, Haskell, Pascal, ML, ADA, APL, C, Visual C++, Visual J#, Visual Basic, Visual C#, SmallTalk, Oberon, Scheme, Mercury, Oz and CAML. See Appendix C for further information on supported languages.



Simplified Deployment

Because CLR applications are self-describing and require no registry entries, you can use XCOPY or FTP to simply copy the application to an appropriate directory so the application can be run from that directory.

Use code download - If you are distributing your application on the Internet or through a corporate Intranet, you can simply download the code to a computer and run the application there.

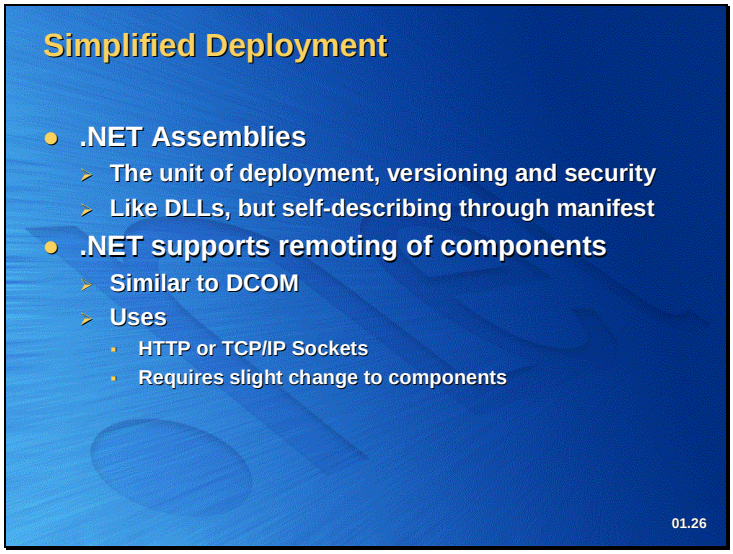
Use an installer program such as the Windows Installer 2.0 - The Windows Installer 2.0 can install, repair or remove Microsoft .NET Framework assemblies in the global assembly cache and in private directories.

Note: In order to run a .NET application, you must have .NET CLR installed on the client.

Deployment scenarios can be found at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguidnf/html/cpcondeploymentscenarios.asp

 ©2002 Microsoft Corporation

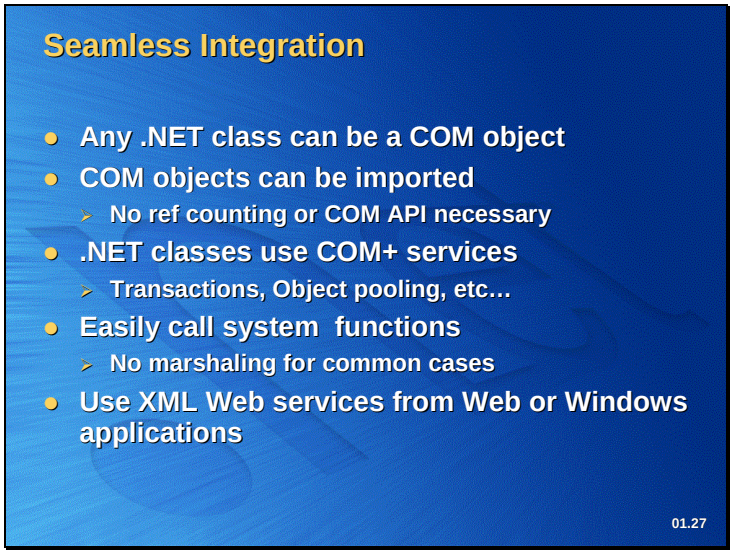


Simplified Deployment

.NET Assemblies are the primary building block of a .NET Framework application. An assembly is a collection of functionality that is built, versioned and deployed as a single implementation unit (in one or more files). All managed types and resources are marked either as accessible only within their implementation unit or as accessible by code outside that unit. Assemblies are self-describing by means of their manifest, which is an integral part of every assembly.

Remote objects can be registered either programmatically or by specifying the remote objects in configuration files. Registering remote objects in configuration files allows administrators of applications that expose or consume remote objects to change the location of the objects without recompiling the application. This section describes how to register remote objects in configuration files.

Developers publishing or consuming .NET remoting objects are responsible for adding these objects to the configuration file. Administrators can then use the .NET Framework Configuration tool to point remoting clients at different endpoints or modify the lifetime of client-activated objects where necessary.


Seamless Integration
Any .NET class can be a COM object
COM objects can be imported
No ref counting or COM API necessary
.NET classes use COM+ services
Transactions, Object pooling, etc…
Easily call system functions
No marshaling for common cases
Use XML Web services from Web or Windows applications
01.27

Seamless Integration

Microsoft understands that you're not going to get rid of your existing code, so we made .NET able to seamlessly interoperate with existing code. This interchangeability is vital because you are not forced to rewrite everything. In fact, you can use any Visual Basic .NET class from within Visual Basic 6.0, and, more importantly, you can use any of your existing COM objects from within Visual Basic .NET.
The runtime requires additional attributes to enable usable interop from Visual Basic 6.0 and other ActiveX development tools. Visual Basic .NET makes this really easy with our <ComClass> attribute that you can put on a class instead of having to use a few attributes with GUIDs on every member in that class.

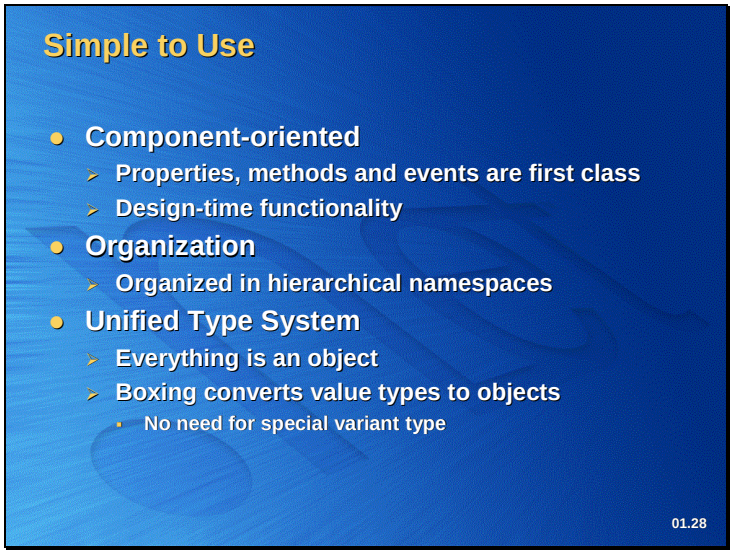
The .NET Framework is a natural progression from COM since the two models share central themes including component reuse and language neutrality. For backward compatibility, COM interoperability provides access to existing COM components without requiring modification of the original component. A .NET Framework developer can incorporate COM components into a managed application by using COM interoperability tools to import the relevant COM types. Once imported, the COM types are ready to use.

The Microsoft .NET Framework promotes interaction with COM components, COM+ services, external type libraries and many operating system services. DataTypes, method signatures and error-handling mechanisms vary between managed and unmanaged object models. To simplify interoperation between the .NET Framework components and unmanaged code and to ease the migration path, the CLR conceals from both clients and servers the differences in these object models.

Code executing under the control of the CLR is called managed code. Conversely, code that runs outside the CLR is unmanaged code. COM components, ActiveX interfaces and Win32 API functions are examples of unmanaged code.

SOAP Toolkit interoperability enables you to use XML Web services from Web or Windows applications

 ©2002 Microsoft Corporation



Simple to Use

The **.NET Framework** makes development tasks easier since it is component-oriented. It affects not only how you write code for the CLR, but also makes it easier for tools to provide a rich design-time experience. Object Orientation features of the .NET languages provide a consistent API throughout all the .NET tools, which use common design patterns and hierarchical namespaces. Many of the benefits of the CLR apply to the Framework class libraries as well.

Hierarchical namespaces make it easier to discover where code is located. Contrast this with Windows API, which is flat and it can be difficult to find a particular piece of functionality. A hierarchical namespace within the system makes it much easier for you to explore and discover functionality.

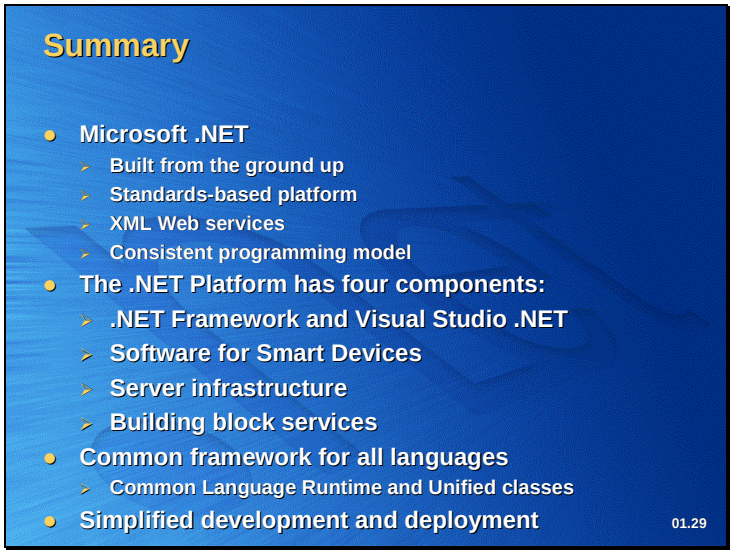
Namespace is a logical naming scheme for grouping related types. For example, the .NET Framework uses a hierarchical naming scheme for grouping types into logical categories of related functionality, such as ASP.NET application framework, or remoting functionality. Design tools can use namespaces to make it easier for developers to browse and reference types in their code. The concept of a namespace is orthogonal to that of an assembly: A single assembly can contain types whose hierarchical names have different namespace roots and a logical namespace root can span multiple assemblies. In the .NET Framework, a namespace is a logical design-time naming convenience, whereas an assembly establishes the name scope for types at run time.

The .NET Framework has a unified type system where everything is an object. This means the death of variants because **variants are now objects.** There is one string type throughout the entire framework. All character data is unicode, meaning it supports all character sets including sets that cannot be represented in a single byte.

Boxing is the conversion of a value type instance to an object, which implies that the instance will carry full type information at run time and will be allocated in the heap. The Microsoft Intermediate Language (MSIL) instruction set's box instruction converts a value type to an object by making a copy of the value type and embedding it in a newly allocated object. For example:

```
123.toString()…
```

In this case, we are boxing the integer 123 into a reference type object so we can call the toString method.



Summary

.NET is a technology evolution built from the ground up for the Web. --It is not a bolt-on solution. It is a new framework that can provide seamless integration of multiple applications and devices. It can be used to take advantage of the abundant computing and communications resources provided on the Internet to enable smart, service-aware systems and richer user experiences.

.NET is based on standards and a unified programming model. Web standards are open and non-proprietary, they are built to be first-class constructs and they drive interoperability and integration of new existing systems. .NET is multi-language because it works in the language of your choice.

XML Web services enable you to build and operate Web services that deliver software as a service. XML Web services are today's best-of-breed business integration since they allow customer centricity in your business applications.

Consistent programming model with the highest productivity tools. Visual Studio .NET is the next generation of the Microsoft Integrated Development Environment. Everyone who has been using Microsoft development tools will feel right at home with Visual Studio .NET.

Note: There is a .NET runtime for Windows 98 and up. The development environment is Windows 2000 and up.

The **.NET Platform** is a set of development tools and operational systems used to build, expose and consume XML Web services. It has four components:

- .NET Framework and Visual Studio .NET - The Integrated Development Environment and languages
- Software for Smart Devices - The Mobil Internet Toolkit is a set of programming interfaces that enable developers to target mobile devices like smart phones and PDAs
- Server infrastructure - MSMQ, COM+, Active Directory, IIS and WMI
- Building block services - Existing helper services like Microsoft Passport™

The CLR manages code at execution time, providing core services such as memory management, thread management and remoting, while also enforcing strict safety and accuracy of the code. Because of the

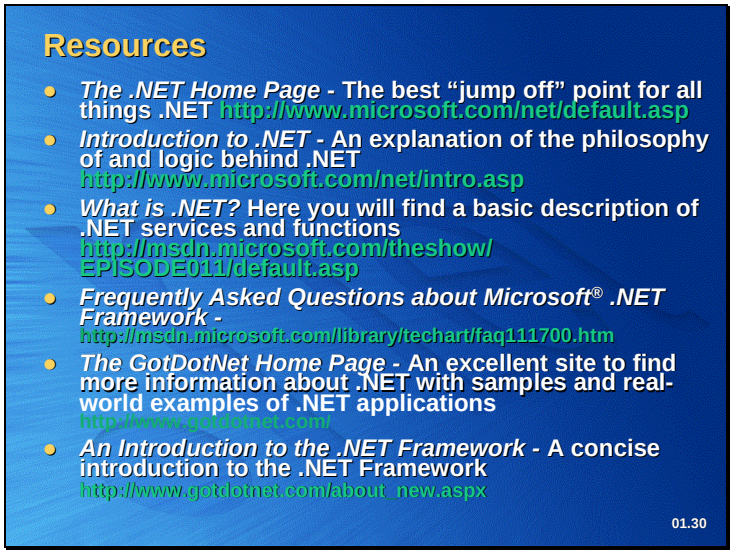
 ©2002 Microsoft Corporation

CLR, objects written in different languages can communicate and their behaviors can be tightly integrated.

The Unified Class Library is a comprehensive, object-oriented collection of reusable classes you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET and Web services.

The .NET Framework is based on standards and it's XML to the core. It provides a common framework for all languages with common UI elements (Windows Forms and Web Forms). The .NET Framework is one quarter of the .NET Platform. The .NET Framework has been built from the ground to provide a new computing platform designed to simplify application development and deployment in the highly distributed environment of the Internet.

The primary focus of the seminar is the .NET Framework, its tools, the Common Language Runtime and Type libraries, the .NET Languages and how they are used to create XML Web services and, Windows and Web Applications that run on .NET. We will also show how the server infrastructure and building block services fit in the .NET framework.


Resources
The .NET Home Page - The best "jump off" point for all things .NET http://www.microsoft.com/net/default.asp
Introduction to .NET - An explanation of the philosophy of and logic behind .NET http://www.microsoft.com/net/intro.asp
What is .NET? Here you will find a basic description of .NET services and functions http://msdn.microsoft.com/theshow/EPISODE011/default.asp
Frequently Asked Questions about Microsoft® .NET Framework - http://msdn.microsoft.com/library/techart/faq111700.htm
The GotDotNet Home Page - An excellent site to find more information about .NET with samples and real-world examples of .NET applications http://www.gotdotnet.com/
An Introduction to the .NET Framework - A concise introduction to the .NET Framework http://www.gotdotnet.com/about_new.aspx
01.30

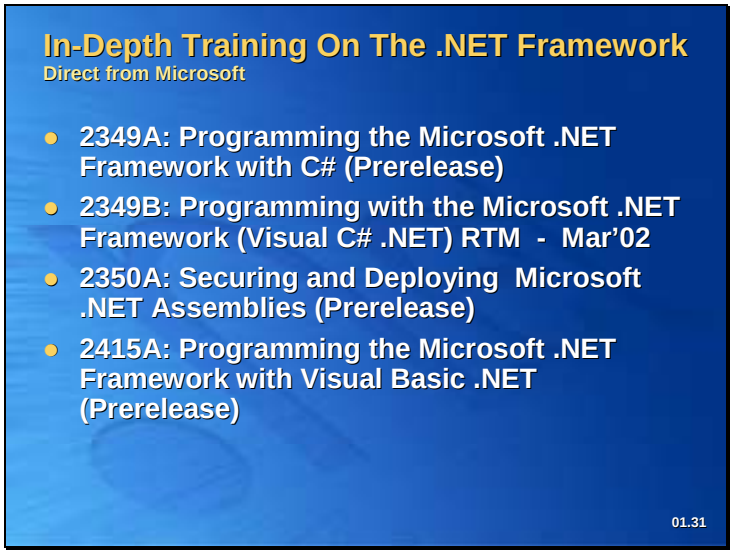
Resources

The .NET Home Page - The best "jump off" point for all things .NET:
http://www.microsoft.com/net/default.asp

Introduction to .NET - An explanation of the philosophy of and logic behind .NET
http://www.microsoft.com/net/intro.asp

What is .NET? Here you will find a basic description of .NET services and functions:
http://msdn.microsoft.com/theshow/EPISODE011/default.asp

 ©2002 Microsoft Corporation


In-Depth Training On The .NET Framework
Direct from Microsoft
2349A: Programming the Microsoft .NET Framework with C# (Prerelease)
2349B: Programming with the Microsoft .NET Framework (Visual C# .NET) RTM - Mar'02
2350A: Securing and Deploying Microsoft .NET Assemblies (Prerelease)
2415A: Programming the Microsoft .NET Framework with Visual Basic .NET (Prerelease)
01.31

In-Depth Training On The .NET Framework
Direct from Microsoft

 ©2002 Microsoft Corporation